International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 4, Issue 1, Feb 2014, 43-56

AN ADAPTIVE DESIGN PATTERN TO BUILD MULTI-MODAL SYSTEM ARCHITECTURE IN DISTRIBUTED ENVIRONMENT

ASHISH PATIL[1], MANJUSHA JOSHI[2], S. D. JOSHI[3] & R. M. JALNEKAR[4]

[1,2]Research Scholar, Department of Computer Engineering, BVDUCOE, Pune, Maharashtra, India

[3]Professor, Department of Computer Engineering, BVDUCOE, Pune, Maharashtra, India

[4]The Director, Vishwakarma Institute of Technology (VIT), Pune, Maharashtra, India

ABSTRACT

Within the last two decades, the World Wide Web (WWW), more commonly known as the web, and it is Distributed System Architecture (DSA) which is the collection of independent system with own resources but users experience as a single system; it is totally integrated system instead of several independent systems, and it has ability to autonomously manage each system. Hence it provides optimum functionality and flexibility. WWW has become the main platform for deploying business and social application and organizational information systems. Many organizations have extended the scope of their web-based systems as well as begun to provide mobile and wireless access to them. Therefore, web-based applications now present an array of content and functionality to a huge number of users and carry out many different purposes.

There are so many web applications based on multimodal interaction are available, which provides different types of multimodal and content services like video/audio chat, text chat services, desktop sharing services and mobile related services (e.g. SMS services). Distributed Multimodal System (DMS) includes the multimodal interaction. Multimodality refers to the process in which different devices and user are able to interact aurally, visually, by touch or by gesture. Ubiquitous computing refers to this interaction happening anywhere, anytime, using any device, often in order to increase the accessibility and improve the user experience.

Modeling multimodal interaction is very difficult, due to the multiple input and output channels, modes and the combination of possibilities between data coming from different sources, not to mention output modality selection based on context and user profile. The huge challenge of multimodal interface creation is to build reliable processing in the multimodal system to analyze and to understand multiple communications means in real-time. There is need to specify a suitable pattern to create a platform which is able to provide multimodal interactions between user with multiple I/O modalities and several semantic services developed by third-party agent and consumed through a presentation, in order to make content more accessible and interactive; in distributed environment.

In WWW, Each one application is only dominant on particular type service as well as most of them are executed on desktop based environment. Hence, these web applications are building with different types of design patterns which are having with its own design methodology to build a particular web application. The general purpose of this research is to develop the applications having with an adaptive design pattern which provides multimodality and deployment of multimodal, content and ubiquitous services in efficient manner for people with distributed system (e.g. Internet). Hence, designing multimodal approach, along with the traditional computing environment or with the ubiquitous environment, is always a challenging task in the HCI field.

Hence we discover a service-oriented architectural (SOA) pattern is the best solution for modeling the DMS. Service-oriented architecture presents an approach to build the distributed system that deliver application functionality as web services to either end-user applications or other services. Applications must be developed as independent sets of interacting services offering well-defined interfaces to their potential users. Similarly, supporting technology must be available to allow application developers to browse collections of services, select those of interest, and assemble them to create the desired functionality.

We have presented a service oriented approach to build the DMS by using an adaptive design pattern Model-View-Controller (MVC) which managing all data and interactions among the users, the system and the external services. Hence the SOA implementations continue their egression in application development for integrating organizational operations in different ways as well as for elevating reuse while leveraging the existing value of legacy systems. It means at the application level, the online presenter can deliver the presentation to attendees which are the part of conference and the interaction between them with multiple I/O modalities such as audio/video chat, text chat as well as desktop sharing services.

KEYWORDS: WWW (World Wide Web), **DSA** (Distributed System Architecture), **DMS** (Distributed Multimodal System), **HCI** (Human-Computer Interaction), **SOA** (Service-Oriented Architecture), **MVC** (Model-View-Controller)

INTRODUCTION

In today's world practice most interactions with computers take place using multiple I/O modalities. Input modalities are event-based or streaming based and it requires a user device to transfer human output into a form suitable for computer processing. Event-based input modalities—such as input via a keyboard or mouse—react to user actions by producing discrete events. Streaming-based modalities sample input signals with some resolution and frequency, producing a time-stamped array of sampled values. For example, a computer detects a user's voice or psychological signals by sampling input signals with sensors such as a microphone. The interaction and the interface between humans and computers are not very natural yet; although they are often lightly multimodal (considering, for instance, the mouse or other pointing devices alongside the keyboard). The appearance of touch screens improved user's navigation through the graphical user interface, but did not introduce a real multimodal interaction between users and computers. Multimodal interactions and multimodality refer to the process in which different devices and people are able to interact aurally, visually, by touch or by gesture. Ubiquitous computing refers to increase the accessibility and improve the user experience.

Multimodality is applied to present usable and accessible information to users, including videos, images and texts. One of the main purposes of multimodality is the improvement of user interactions with devices, such as smart-phones. Information access and interaction for users with functional diversity, is another main purpose of multimodality. User interfaces should allow users to interact with the content or a service through the most appropriate modes, taking into account user's preferences and context.Interactions can be defined as information exchange among people, technologies (represented by user interfaces) and processes. The user may determine the mode or modes of interaction that he prefers for accessing information more naturally. Multimodality extends and improves user interfaces because it allows the integration of voice, image and other types of data input such as keyboards, mice, webcams, pens, touch screens and remotes. The motivating scenario behind the design and development of the multimodal platform with semantic services is one in which people interact with application and confidently interoperate with the web application services in a multimodal way.

Multimodal systems are computer systems empowered with multimodal capabilities for human/machine interaction and able to interpret information from various sensory and communication channels. Literally, multimodal

interaction offers a set of "modalities" to users to allow them to interact with the machine. Multimodal interface processes two or more combined user input modes (such as speech, pen, touch, a manual gesture, head and body movements) in a coordinated manner with multimedia system output. Two important features of multimodal architectures and processing are: (1) the fusion of different types of data; and (2) real-time processing and temporal constraints imposed on information processing. Our aim to reduce the complexity associated with building multi-modal systems; hence there is need to build an adaptable multi-agent architecture for multimodal systems and provide a suite of general purpose, plug-in agents to handle essential tasks like speech I/O, fusion and semantic analysis.

Architecture of Distributed Multi-Modal System

The Distributed Multimodal System (DMS) contains the major software component such as interaction manager, multimodal server and service broker(s).And also the generic components for handling of multimodal integration are: a fusion engine, a fission module, a dialog manager and a context manager, which all together form is Task manager.

- **Interaction Manager (IM) :**The IM's role is the management of the user interactions between the web application and the user. Figure 1 illustrates the processing flow between these components, the input and output modalities, as well as the potential client applications in which input modalities are first perceived though various recognizers, which output their results to the fusion engine, in charge of giving a common interpretation of the inputs.

- **Task Manager (TM) :** The main goal of the Task Manager (TM) is to manage multiple users. When the fusion engine comes to an interpretation, it communicates it to the dialog manager, in charge of identifying the dialog state, the transition to perform, the action to communicate to a given application, and/or the message to return through the fission component. The fission engine is finally in charge of returning a message to the user through the most adequate modality or combination of modalities, depending on the user profile and context of use.The User Modeller is another component of the model. The user interface is generated with the User Interface Generator (UI Generator) module, which would be the view for user. The TM has all the user profiles serialized, and knows the context and the user's preferences which are managed by the User Modeller, in order to adapt the system and the content according to user's necessities and context. The TM creates a task, generating an instance of the user's context in order to inform the IM about the user's preferences, and the IM adapts the content and integrates it with multimodality for the user. The TM is able to get the content of third-party distributed services in order for the IM to generate the user interface. The content will be obtained through the Service Broker which would manage Semantic Web Services.

- **Multimodal Server (MS) :** The MS is included into the controller modules with the TM, indirectly with IM and it is able to manage all third-party distributed services in order to process and synthesize human-computer interactions. The main goal of this module is to provide communication, in the experimental platform, among the interface and the multimodal services able to understand the user and communicate with each other. It provides uncoupling between the services and the IM, as well as the reusability of different services from different parties without knowing who is providing the service.

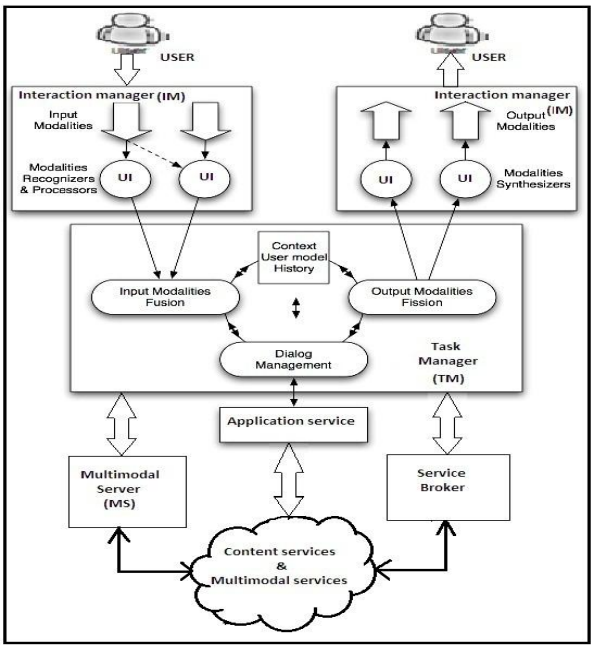

Figure 1: Architecture of Distributed Multi-Modal System (DMS)

- **Multimodal Services**

Multimodal services are managed directly by the MS. Distributed, non-local services currently available in the experimental platform For example voice service - if the user is interacting with the system using the voice, the voice recognition process will know how to connect with the user's device because the MS sends it a message with the IP address, port and other security parameters. All messages among all services and system processes are exchanged using extensible mark-up language based language.

- **Content Services**

Content services are third-party, semantic, Web services (e.g. city services, weather services) managed through the Service Broker, which will be in charge of discovering and requesting new services, and adapting the content depending on the context, user preferences and IM requests.

Applications Domain of Multi-Modal System in Distributed Environment

Multimodal interfaces offers flexibility providing multiple channels for interaction. The main idea of the experimental platform developed is to manage distributed services that offer the capability of processing and synthesizing the multiple modalities of interaction of the user interface.

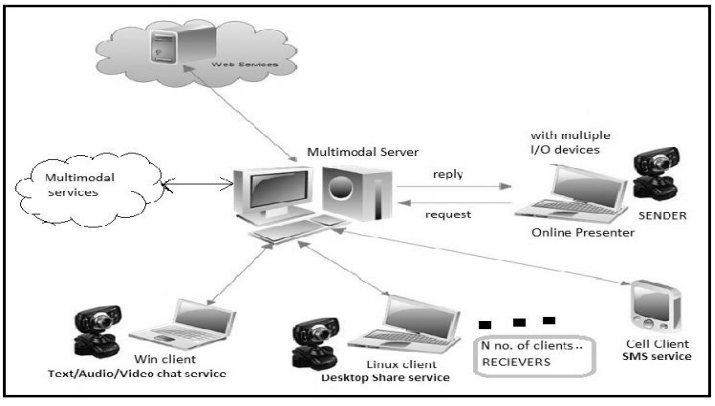

Figure 2: Example of Distributed Multi-Modal System

These services are enriched with semantics and adapted to user necessities through an adaptor and a service bus which manage the multimodal, distributed, semantic services depending on user's preferences, necessities and modelling. For example, an online meeting is a meeting in which the presenter connects to the attendees through an application in distributed environment. This option allows the presenter and attendees to share video, audio, programs, instant messages and shared desktop.The system architecture indicates the flow of the overall system where presenter and multiple clients can connect with valid IP provided by multimodal server. If the connection set-up successfully, then it is the sign of successful configuration between server, presenter and clients. Hence the presenter and multiple clients can share different types of services such as audio/video chat, text chat as well as desktop sharing simultaneously like web service application environment. . So there is demand of web services such as audio & video service, text chat service and SMS service.

PROBLEMS DOMAIN IN DISTRIBUTED MULTIMODAL SYSTEM (DMS)

Software development becomes more and more a global process that involves project teams from all over the world working in concert to achieve a common goal. It means the development of a single software product may involve project teams from different sites. Thus, the new challenge is the collaboration and management of the development teams at multiple remote sites which work together for the same project, and DMS is in collaborative type system. Hence, Building a distributed multimodal system is a complex undertaking in mobility and Ubiquitous environment. Because of its ubiquitous presence, the expectations and demands placed on the web applications have increased significantly over the years. At the same time, the development, deployment and maintenance processes of the web-based systems which have become more and more complex and difficult to manage, have not progressed at a sufficient rate to meet these demand and challenges. With existing system, there is difficulty in the interoperability and portability along with technology which is to be used and problem with compatibility of the web browser which is used at client side.

In the above example (online meeting) of DMS, how an online videoconferencing environment with its multiple modalities can be used in language presenting; can the presenter and attendees adapt to the multimodal online environment and how new patterns of communication emerge in the process. Hence, information and communication technology system suffers from an inability to satisfy the heterogeneous needs of many users. The problems of heterogeneity, interoperability and ever changing requirements, such as the architecture provide a platform for building application services with the following characteristics – Loosely coupled, location transparent and protocol independent. In DMS, there is different software agents involved in the interaction between different services, so there is huge coupling among these software agents. And also different applications with different technology implementation endure to coordinate the application services.

PROPOSED SOLUTION

We presents a Service-oriented architectural (SOA) approach for building distributed multimodal systems that deliver application with better functionality and delivers services to either end-user applications or other services; with quality of service where a service is a unit of work provided by a service provider to achieve desired results for a service consumer. The objective of SOA is to reduce coupling among interacting software agents. In a distributed application system, in many cases the complex between objects are many communication relations, and therefore in the development process we utilize design patterns on the complementary advantages of each phase and reward each other to enhance the overall stability of the system architecture and robustness.

Design pattern plays an important role in SOA implementation, especially to provide standardization of service design; and also speed up the development process through the implementation of tried and tested solutions. Patterns are

about design and interaction of objects, as well as providing a communication platform concerning refined, reusable solutions to commonly encountered programming challenges. There are so many design patterns are used in the service-oriented architecture; but our research findings the model-view controller (MVC) is the good solution to build DMS with service oriented approach. The model-view controller (MVC) pattern of service oriented approach managing all data and interactions among the users, the system and the external services in DMS.

Service-Oriented Architecture (SOA)

SOAs provide modular services that can be easily integrated throughout collaborative managerial services. They are flexible and adaptable to the current information technology development. Many SOA implementations use Web services. A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model. [15]

Service-oriented computing (SOC) in system engineering is a new paradigm that divides the systems into basic entities. [14]

- **Service:** A logical entity; the contract defined by one or more published interfaces.
- **Service provider:** The software entity that implements a service specification.
- **Service requestor:** The software entity that calls a service provider. Traditionally, this is termed a “client”; now here the service requester can be an end-user application or another service.
- **Service locator:** A specific kind of service provider that acts as a registry and allows for the lookup of service provider interfaces and service locations.
- **Service broker:** A specific kind of service provider that can pass on service requests to one or more additional service providers.

Service-oriented architectural businesses and their processes are implemented through Web services and their interactions. Web services interactions can be distributed across applications and enterprises. The successful implementation of SOA depends on a measured and emphasizing the functional relationship approach to business planning. Service-Oriented Architecture is envisioned most efficient in designing enterprise systems and integrating heterogeneous applications in a distributed environment as well as it enables complex systems to be divided into loosely coupled services. The most essential part of SOA is that it differentiates the service's implementation from its interface. So Web services technologies make system components to be interoperable and extensible within an enterprise as well as across enterprise boundaries. An additional consideration is that operational systems will typically be distributed across many machines to improve performance, availability, and scalability. An enterprise solution has to coordinate functionality executing on a collection of hardware.

One way to conceive of such a system is to consider it to be composed of a collection of interacting services. Each service provides access to a well-defined collection of functionality. The system as a whole is designed and implemented as a set of interactions among these services. Exposing functionality as services is the key to flexibility. A system evolves through the addition of new services. The resulting service-oriented architecture (SOA) defines the services of which the system is composed, describes the interactions that occur among the services to realize certain behaviour, and maps the services into one or more implementations in specific technologies.

MVC: A Design Pattern For DMS Development

Service-oriented architecture renders the Model-View-Controller (MVC) architecture where the controller is a key component, used as an intermediate between the user interface (UI) and the application data.

Basics Components of MVC model as follows-

- **Model :** the model contains the data that the application requires that means it encapsulates the business logic & processing.
- **View :** the view manages the user interface (UIs).
- **Controller :** It provides the logic and serves as the interface between the model and the view, it means manages navigation & input.

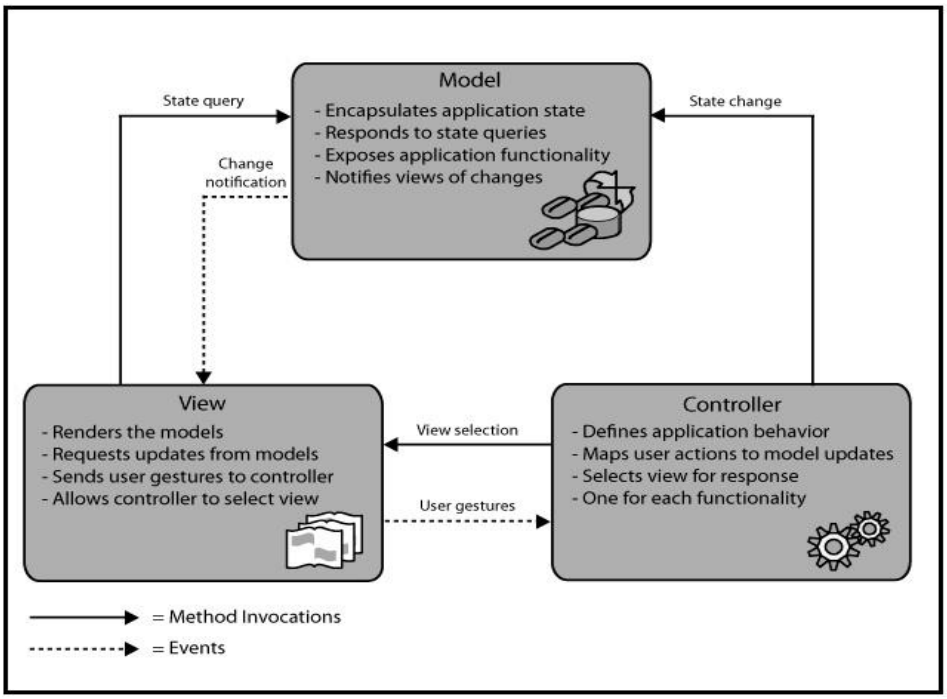

Figure 3: Model View Controller (MVC) Architecture

As a figure 1 illustrates the Distributed Multimodal System (DMS) architecture, where we apply a model-view controller (MVC) design pattern which managing all data and interactions among the users, the system and the external services. As a MVC pattern component, the task manager (TM) is the model, because it interacts with the interaction manager (IM), making transparent all the accesses to services, the message transformation, and the protocol used by the multimodal server MS. The IM is the controller into a MVC pattern and it is the core of the architecture and its role is the management of the user interactions via user interface (View).

Benefits of Proposed System

- **Resource Accessibility and Reusability:** Multimodal system in distributed environment which support user access to remote resources (e.g. printers, data files, web pages, and CPU cycles) and the reusability of application service provides the economical sharing of resources. So it is beneficial to enhance reusability and maintainability of the system that reflects the many to many communication relationships.

- **Multimodality:** The proposed system having with multiple input and output channels and modes accessibility. So it provides multiple user interaction options to use the application services
- **Distribution Transparency:** Our system hides the fact from user-side from where the application service gets delivered and provides a single system image to the users to interact with the system.
- **Openness:** The system provides multiple interfaces which are openly accessible to empower simple extensions to existing components and include new components. So it has functionality to support interoperability, portability, extensibility are the features of good quality of services (QoS).
- **Scalability:** The system has a highly scalable capacity because of distributed environment it means scalability with respect to size (number of users), geographic distribution, administrative domains.
- **Performance Enhancement:** As per discussion, the DMS system provides multimodality, mobility, accessibility, interoperability, extensibility, flexibility and reliability, hence the overall system performance is enhanced.

BACKGROUND & MOTIVATION

What is Distributed System-?

Distributed systems are collection of independent computers or group of networked computers which have the same objective for their function.This kind of system has the following characteristics -

- **Connecting Resources and Users is A Means Of Resource Sharing and Load Balancing:** It provides efficient and responsive resource utilization.
- **Distribution Transparency:** It is to make the existence of multiple computers invisible (transparent) and provide a single system image to its users.
- **Scalability:** It refers to the capability of a system to adapt to increased service load.
- **Reliability, Availability and Fault Tolerance:** It achieved through redundancy and dynamic allocation.
- **Enhanced Performance Potential:** It derived from parallel operation in huge distributed system.

Distributed system is classified into two types -1) Homogeneous distributed system 2) Heterogeneous distributed system.

In homogeneous DS, all sites they have identical software as well as hardware configuration and it appears to user as single system. All sites are known to each other so that they agree to co-operate with user request.

In heterogeneous DS, different sites use different schemas of hardware, platform and language. So it may not be aware of each other & they provide only limited facilities for co-operation in transaction processing.

Architecture of Distributed System

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is made up of loosely-coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system. Distributed programming typically falls into several basic architectures viz. Client-server, 3-tier architecture, N-tier architecture, distributed objects, loose coupling or tight coupling.

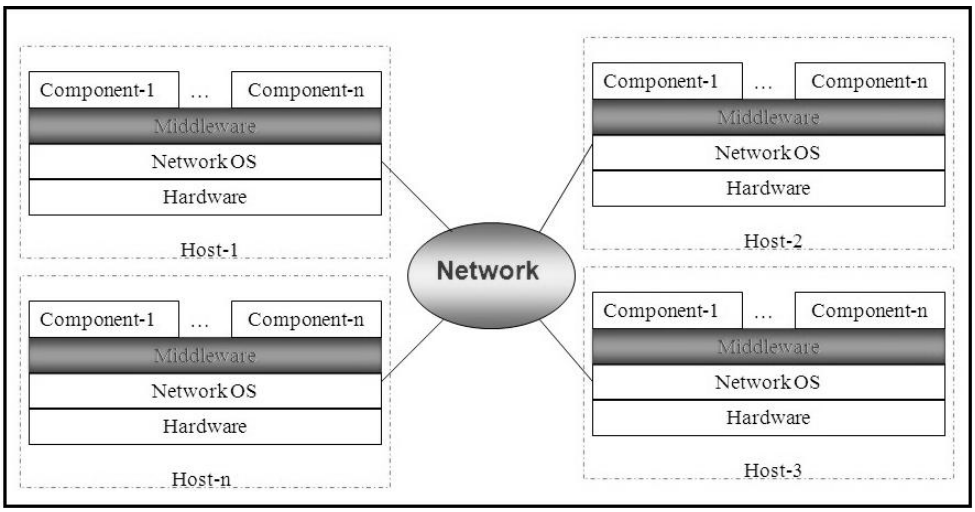

Figure 4*:* Distributed System Basic Architecture

Distributed system architecture consist of the following components as shown in the figure 4.

- **Computer Workstations (Sites Or Nodes):** A distributed DBMS consists of a number of computer workstations that form the network of system. The distributed database system must be independent of the computer system hardware and situated across the nodes which are participants in network.
- **Network Components (Both Hardware and Software):** Each workstation in a distributed system contains a number of network, hardware and software components. These components allow each site to interact and exchange data with each other. Network system independence is a desirable property of the distributed system.
- **Communication Media:** In a distributed system, any type of communication (data transfer, information exchange) among nodes is carried out through communication media. This is a very important component in a distributed management system. It is desirable that a distributed DBMS be communication media independent, that is, it must be able to support different types of communication media (e.g. wired or wireless communication).
- **Middleware:** Too many networked applications existed hard or difficult to integrate due to departments are running on different platforms (NOSs). So middleware plays an important role in integration and interoperability only at primitive level NOS services.
- **Transaction Manager (TM):** A TM is a software component that resides in each computer connected with the distributed system and is responsible for receiving and processing both local and remote applications data requests.
- **Data Manager (DM):** A DM is also a software component that resides in each computer connected with the distributed system and stores and retrieves data located at that site. In a distributed DBMS, a DM may be a centralized DBMS.

Demand of Distributed System

A distributed system may have a common goal such as solving a complex and large computational problem as well as connecting resources to improve its utilization. There are genuine benefits in building distributed systems are.

- **Distributing Machines May Make Resource to Another:** Each computer may have its own user with individual needs and to coordinate the use of shared resources or provide communication services to the users.

- **Co-Operative and Social Networking:** Clients that are geographically differentiated can now work and play together. Example of this scenario is - distributed document systems, remote desktop sharing or handling PC, audio or video conferencing, messaging, multiplayer games and social networks.
- **Increased Reliability:** Assuming that a small percentage of machines break; whatever remains of the framework are done through another participants in that framework and also can do useful work. Hence the system removes failures in individual computer system in network.
- **Incremental Growth:** It has delivery aided capacity when and where needed. The structure of the system is not known in advance which includes network topology, network latency, number of computers etc. The system may consist of different kinds of computers, network links and the system may change during the execution of a distributed program. Networking allows permits you to add onto an existing infrastructure.
- **Remote Access and Services:** User may need to access information held by others at their systems. Examples of this include web browsing, remote file access, and programs to retrieve large files.
- **Mobility:** Clients move around with their devices. So it is not feasible for them to carry all the qualified information they require with them.

RELATED WORK

Software design patterns are best practice solutions to common software problems. There are numerous eminent approaches for software architectures from design patterns and are in particular real time systems. These approaches just give unique descriptions of design patterns and high level guidance on how the patterns can be used to form software architectures. [3]

Distributed systems can be perceived as living organisms in the sense that the state of the computing system as well as its execution environment conditions change dynamically. In order to provide the intended services and functionalities at the required quality of service, adaptation of the system to the changing execution environment is necessary. Adaptive systems that can change their behavior at run time have a number of potential benefits. For example, adaptive distributed systems can respond rapidly to improve the opportunities to optimize performance as the execution environment changes. [11]

To build distributed system & web applications the middleware methods & design methods are used. The middleware methods consist of communication, content & business process levels. At communication level, there are technologies that support tightly or loosely coupled communication styles. The various design methods are specifically proposed for concurrent & distributed systems. Due to their support for modularity, flexibility & reusability the object oriented methodologies have been proposed for the design of DSA. [3]

Patterns are defined as "an idea that has been useful in one practical context and will probably be useful in others". The Patterns for e-business are a group of proven, reusable assets that can be used to increase the speed of developing and deploying Web applications. There are various design patterns like adapter, façade, proxy and controller, can be used to implement enterprise business web applications using the service-oriented architecture approach. [14] [15] The experimental platform is demonstrate that is possible to offer multimodal interfaces in a distributed architecture responsible of the sensory input and output services, directed through a multimodal and interaction manager. Multimodal Architecture and Interfaces, improves some of its runtime framework sub-components and explores connecting the modality components to the framework as a remote service. [9]

ISSUES & CHALLENGES

Different kinds of distributed systems work today, every pointed at tackling various types of issues. The challenges faced in building a distributed multimodal system vary depending on the requirements of the system. In general, on the other hand, most systems will need to handle the following issues. [11] [12]

- **Heterogeneity:** Different subsystem in the system must have the capacity to interoperate with each one in turn, despite distinctions in hardware architectures, operating systems (platforms), communication protocols, programming languages, software interfaces, security models and data formats. Due to multiple types of heterogeneity which poorly support for maintaining global consistency of data stores.
- **Multimodality:** The multiple input/output channels and modes of accessibility it means combination of possibilities between data coming from different sources. Hence there is the complexity associated with building multi-modal systems.
- **Transparency:** The whole framework should appear as a single unit and the complexity and interactions between the components should be typically hidden from the end user.
- **Fault Tolerance and Failure Management:** Failure of one or more components should not bring down the whole framework or entire system, and should be isolated. Replication of data and services provides fault tolerance and availability, but at a cost.
- **Scalability:** The system may as well work effectively with expanding number of clients and expansion of a resource may as well enhance the performance of the system. Explicit and tedious programming related with data and network of applications on each device.
- **Concurrency**: Shared access to resources should be made possible. So there is need of advanced software to realize the potential benefits.
- **Openness and Extensibility:** Interfaces should be cleanly differentiated and openly accessible to empower simple extensions to existing components and include new components. But Poor middleware (Interfaces) support e.g. dis-connectivity, location independence, group collaboration, atomic transaction, quality of services (QoS).
- **Migration and Load Balancing:** Permit the development of undertakings inside a framework without influencing the operation of clients or applications and distribute load among available resources for enhancing performance. But it affects on multiple network topology, bandwidth and Latency problem
- **Security:** Security and privacy concerns regarding network communication. Access to resources should be secured to guarantee just known clients have the ability to perform permitted operations.
- **SOA Building Challenge:** As SOA services are typically coarse-grained and loosely coupled; their operations expose more latency than more tightly coupled implementations. This can be a challenge when implementing with real-time requirements; and also SOA is designed to bring together legacy systems in heterogeneous environments. [14]

CONCLUSIONS AND FUTURE ENHANCEMENT

In today world practice, the popularity of distributed system applications is increasing day by day. As per discussion, there are many challenges in the distributed management system. So developers are focusing on the

advancements of distributed system applications using various techniques. In distributed multimodal system, most of the interactions with computers take place using multiple I/O modalities. Input modalities are event-based or streaming based and it requires a user device to transfer human output into a form suitable for computer processing. To build a multimodal architecture based on a distributed architecture paradigm using third party services, we need an adaptive design pattern. Design patterns are one of such technology that is used in the design and development of such distributed multimodal architectural applications.

In this paper we have presented a service oriented approach to build the DMS by using an adaptive design pattern Model-View-Controller (MVC) which is the good solution to overcome the problems in designing the distributed multimodal system applications. The SOA approach provides multimodal interaction as well as tools allowing rapid creation of multimodal interfaces. Multimodal Architecture and Interfaces, improves some of its runtime framework sub-components and explores connecting the modality components to the framework as a remote service. The main goal is to create a platform which is based on real time applications to interact with the users over a service-oriented architecture has to improve in some particular features. It means at the application level, the online presenter deliver multiple services such as audio/video chat, text chat and desktop sharing services to the multiple clients concurrently.

Other important future directions for multimodal research include human-machine interaction using new tangible interfaces such as digital paper with pen, multitouch tables surfaces and screens. Further modeling of multimodal interaction still is needed too, in areas such as multimodal educational exchanges, collaborative multimodal interaction, multimodal interaction involving different types of user groups, and mobile multimodal interaction with emerging cell phone applications.Finally, further work is needed to improve tools for the creation of multimodal applications and interfaces so they can become more mainstream, especially since multimodal interfaces are viewed as the most promising line of approach for achieving universal access in the near future.

REFERENCES

1. Marc Pou, Luigi Ceccaroni, Barcelona Digital Centre Tecnològic Barcelona, Spain, “Multimodal Interaction in Distributed and Ubiquitous Computing”, IEEE (computer society 2010).
2. Ronald Strebelow, Mirco Tribastone and Christian Prehofer, “Performance Modeling of Design Patterns for Distributed Computation”, IEEE (computer society 2010).
3. F.Dabous, F.Rabhi, H.Yu, “Using Software Architectures and Design Patterns For Developing Distributed applications”, IEEE (ASWEC-2004)
4. J. S. Fant, “Building Domain specific Software architecture from Software architectural design Patterns”, ICSE-2011.
5. C. Tianhuang, H. Feifei, Department of Computer Science and Technology Wuhan University of Technology, Wuhan, China, ”Design Patterns Application in a distributed system”, IEEE-2010.
6. Sara Maalal, Malika Addou, Team of Systems Architecture, Laboratory of computing, Systems and Renewable Energy National and High School of Electricity and Mechanic ENSEM BP 8118, Oasis Casablanca, Maroc, “A new approach of designing Multi-Agent Systems with a practical sample”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11, 2011.
7. Yinsheng Li, Jianping Shen, Ying Huang, Weiming Shen, “Multi-Model Driven Collaborative Development Platform for Service-Oriented e-Business Systems”, International Journal of Advanced Engineering Informatics

(IJAEI), v. 22, no. 3, July2008, .pp 328-339.

8. Duane Nickul, Laurel Reitman, James Ward, Jack Wilber, “Service Oriented Architecture (SOA) and Specialized Messaging Patterns-2007”.
9. Nicu Sebe, Journal of Ambient Intelligence and Smart Environments-1 (2009) 19–26, “Multimodal interfaces: Challenges and perspectives”, IOS Press – 2009.
10. PetruEles, “Distributed system”.
11. FRED B. SCHNEIDER ON DISTRIBUTED COMPUTING, IEEE DS Online, Volume 1, Number 1 Interview by DejanMilojicic, dejan@spica.hpl.hp.com
12. Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya, “Distributed Systems and Recent Innovations: Challenges and Benefits”.
13. Andy Crabtree, Terry Hemmings, and Tom Rodden, “Pattern-based Support for Interactive Design in Domestic Settings”, ACM-2002.
14. Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, Tony Newling, “Patterns: Service-Oriented Architecture and Web Services”, IBM, April 2004.
15. Mike Rosen, Boris Lublinsky, Kevin T. Smith, Marc J. Balcer, “Applied SOA: Service-Oriented Architecture and Design Strategies”, Wiley Publishing, Inc., 2008.